II. AMENDMENTS TO THE CLAIMS

The following listing of claims replaces all prior versions, and listings, of claims in the application:

1. (Currently Amended) A method of operating a computer system having one or more program classes loaded therein, said method comprising the steps of:

providing a constant pool having constant pool entries for storing data items related to a program class;

performing a resolution on at least some of the data items in the constant pool, said resolution transforming a data item from an unresolved value as loaded in a constant pool entry to a resolved value such that the data item can be utilized by a program; and

maintaining both the unresolved value and the resolved value in the constant pool entry for ~~a resolved~~ the data item, wherein the constant pool entry has an unresolved value field for maintaining the unresolved value and a resolved value field for maintaining the resolved value.

2. (Original) The method of claim 1, further comprising the step of setting a resolution flag for a data item after that data item has been resolved.

3. (Currently Amended) The method of claim 1, wherein the step of performing a resolution for a data item comprises the steps of:

retrieving the unresolved value of the data item;

determining the resolved value of the data item; and

writing the resolved value of the data item into the constant pool entry, whilst still maintaining the unresolved value of the data item in the constant pool entry.

4. (Original) The method of claim 3, wherein the steps of retrieving, determining and writing are performed without locking the constant pool or any component thereof.

5. (Currently Amended) The method of claim 3, wherein the step of performing a resolution for a data item further comprises the step of setting a resolution flag for a data item after the resolved value of that data item has been written into the constant pool entry.

6. (Original) The method of claim 2, wherein a data item within a constant pool has as its unresolved value an index to the name of a class, and said method further comprises the step of always accessing said unresolved value to obtain the class name, irrespective of whether said resolution flag has been set.

7. (Currently Amended) The method of claim 1, wherein said program classes are loaded into a ~~Java virtual machine~~ bytecode environment which includes said constant pool.

8. (Currently Amended) The method of claim 7, wherein the computer system supports a configuration of shared ~~Java virtual machines~~ bytecode environments, such that data items in the constant pool in one ~~Java virtual machine~~ bytecode environment can be resolved in response to processing in another ~~Java virtual machine~~ bytecode environment.

9. (Currently Amended) A computer system having one or more program classes loaded therein, and further comprising:

a constant pool in memory, the constant pool having constant pool entries for storing data items related to a program class;

means for performing a resolution on at least some of the data items in the constant pool, said resolution transforming a data item in a constant pool entry from an unresolved value as loaded to a resolved value such that the data item can be utilized by a program; and

means for maintaining both the unresolved value and the resolved value in the constant pool entry for the data item,

wherein said constant pool entry includes ~~fields~~ an unresolved value field for maintaining ~~both~~ the unresolved value and a resolved value field for maintaining the resolved value ~~in the constant pool entry for a resolved~~ of the data item.

10. (Original) The system of claim 9, wherein the constant pool further includes a resolution flag for each data item, wherein said flag is set after that data item has been resolved.

11. (Currently Amended) The system of claim 9, wherein the means for performing a resolution for a data item comprises:

means for retrieving the unresolved value of the data item;

means for determining the resolved value of the data item; and

means for writing the resolved value of the data item into the constant pool entry, whilst still maintaining the unresolved value of the data item in the constant pool entry.

12. (Original) The system of claim 11, wherein the retrieving determining and writing are performed without locking the constant pool or any component thereof.

13. (Currently Amended) The system of claim 11, wherein the means for performing a resolution for a data item further comprises means for setting a resolution flag for a data item after the resolved value of that data item has been written into the constant pool entry.

14. (Original) The system of claim 10, wherein a data item within a constant pool has as its unresolved value an index to the name of a class, and said system further includes means for accessing said unresolved value to obtain the class name, irrespective of whether said resolution flag has been set.

15. (Currently Amended) The system of claim 9, wherein said program classes are loaded into a ~~Java virtual machine~~ bytecode environment which includes said constant pool.

16. (Currently Amended) The system of claim 15, wherein the computer system supports a configuration of shared ~~Java virtual machines~~ bytecode environments, such that data items in the constant pool in one ~~Java virtual machine~~ bytecode environment can be resolved in response to the processing in another ~~Java virtual machine~~ bytecode environment.

17. (Currently Amended) A computer program product comprising program instructions on a medium which when loaded into a computer system having one or more program classes loaded therein will cause the computer to perform a method comprising the steps of:

providing a constant pool having constant pool entries for storing data items related to a program class;

performing a resolution on at least some of the data items in the constant pool said resolution transforming a data item in a constant pool entry from an unresolved value as loaded to a resolved value such that the data item can be utilized by a program; and

maintaining both the unresolved value and the resolved value in the constant pool entry for ~~a resolved~~ the data item,

wherein the constant pool entry has an unresolved value field for maintaining the unresolved value and a resolved value field for maintaining the resolved value.

18. (Original) The computer program product of claim 17, wherein the program instructions further cause the computer to perform the step of setting a resolution flag for a data item after that data item has been resolved.

19. (Currently Amended) The computer program product of claim 17, wherein the step of performing a resolution for a data item comprises the steps of:

retrieving the unresolved value of the data item:

determining the resolved value of the data item; and

writing the resolved value of the data item into the constant pool entry, whilst still maintaining the unresolved value of the data item in the constant pool entry.

20. (Original) The computer program product of claim 19, wherein the steps of retrieving, determining and writing are performed without locking the constant pool or any component thereof.

21. (Currently Amended) The computer program product of claim 19, wherein the step of performing a resolution for a data item further comprises the step of setting a resolution flag for a data item after the resolved value of that data item has been written into the constant pool entry.

22. (Original) The computer program product of claim 18, wherein a data item within a constant pool has as its unresolved value an index to the name of a class, and said program instructions further cause the computer to perform the step of always accessing said unresolved value to obtain the class name, irrespective of whether said resolution flag has been set.

23. (Currently Amended) The computer program product of claim 17, wherein said program classes are loaded into a ~~Java virtual machine~~ bytecode environment which includes said constant pool.

24. (Currently Amended) The computer program product of claim 23, wherein the computer system supports a configuration of shared ~~Java virtual machines~~ bytecode environments, such that data items in the constant pool in one ~~Java virtual machine~~ bytecode environment can be resolved in response to processing in another ~~Java virtual machine~~ bytecode environment.